intel®

APPLICATION NOTE

AP-89

May 1980

Prototyping with the 8089 I/O Processor

Robin Jigour
Microcomputer Applications

AFN 01153A

Additional copies of this manual or other Intel literature may be obtained from:

Literature Department
Intel Corporation
3065 Bowers Avenue
Santa Clara, CA 95051

Prototyping with the 8089 I/O Processor

Contents

AFN 01153A

INTRODUCTION

Since microprocessors have taken on larger and more sophisticated data processing tasks, it has become a concern to minimize system complexity yet increase performance. Distributed processing is a well accepted solution for this problem. The 8089 I/O Processor is part of this solution for the 8086 microprocessor family. Designed specifically for I/O handling, the 8089 I/O Processor offloads Real Time I/O interfacing from the 8086. The end result provides simplicity, flexibility and increased performance for the overall system.

Designing with the 8089 I/O Processor greatly minimizes system software and hardware efforts. As with most designs however, before approaching the final stages a prototype must be built to evaluate and verify operation. This application note describes such a prototype system for the 8089 I/O Processor. Complete implementation of this system is explained including prototype construction and execution of a demonstration program. Thorough understanding of 8089 and 8086 operation is recommended before using the "8089 Prototype System". In retrospect, various Intel literature may prove useful as reference to this note, this literature includes:

The 8086 Family User's Manual
The 8086 and 8089 Data Sheets
iSBC 86/12A™ Hardware Reference Manual
iSBC 957™ Package User's Guide
iSBC 604/614™ Cardcage Hardware Reference Manual
iSBC 635™ Power Supply Users Manual
MCS86™ Software Development Utilities Manual
PLM/86 Programming Manual
8089 Assembler User's Guide
Universal PROM Programmer User's Manual
ISIS II User's Manual
Intel Multibus™ Specification

THE 8089 PROTOTYPE SYSTEM

The 8089 Prototype System is a basic execution vehicle for 8089 software and hardware within an 8086-8089 remote system configuration. The system consists of various modules shown in block diagram form in Figure 1. The basis of the system is an iSBC 86/12A Single Board Computer and an 8089 Prototype Board. These two boards communicate via the Multibus™ interface on an iSBC 604 cardcage. The 8089 Prototype Board is a remote 8089 system constructed on an iSBC 905 Universal Prototype Board. Power is supplied to the system by an iSBC 635 power supply. Development capability is provided by an Intellec® Microcomputer Development System and an iSBC 957 package. ICE86™ may be used in place of the iSBC 957 package, however it's not specifically used in this application note. The 8089 Prototype Board also provides a serial interface to a CRT terminal which can be used for development.

Serving as an evaluation and development tool, the 8089 Prototype System allows the user to execute 8089 software, establish 8086-8089 initialization and communication protocol, and implement 8089 hardware interfacing. Software for the 8086 and 8089 can be developed on the Intellec Microcomputer Development System and then down-loaded into the 86/12A dual port RAM using the iSBC 957 package. Both the 8086 and 8089 execute code out of the dual

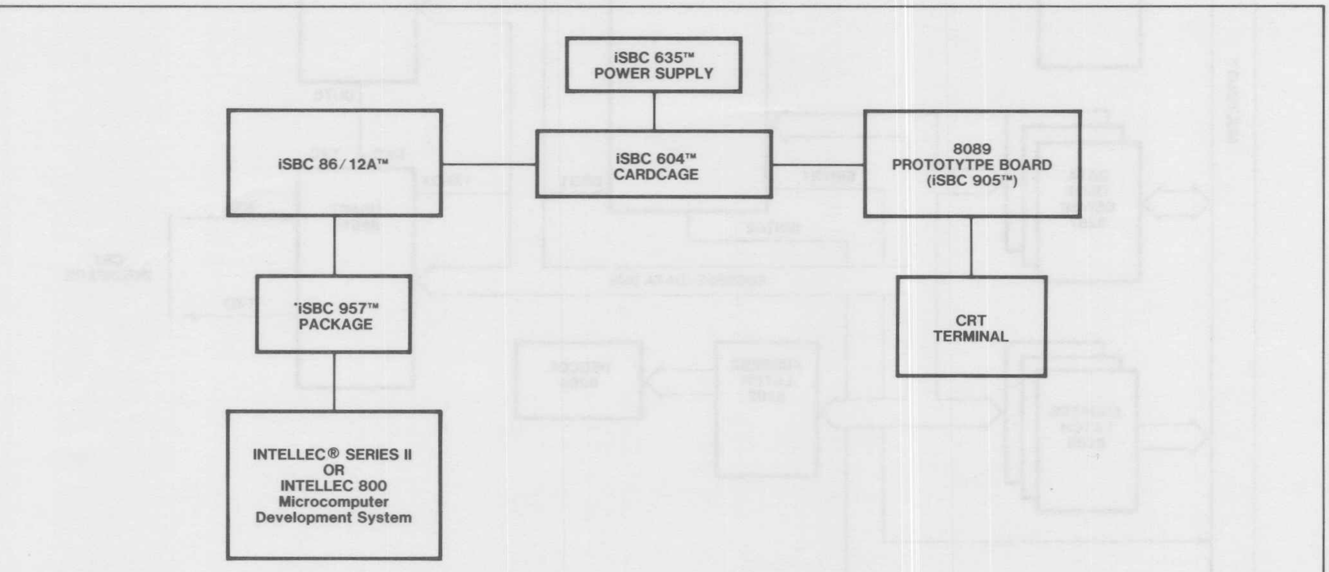


Figure 1. 8089 Prototype System Block Diagram

AFN 01153A

port RAM. Once down-loaded, the program can be run and proper operation verified. Program interrogation can be accomplished using the various commands of the iSBC 957 package monitor. Using these commands, proper protocol for the 8089 initialization and communication blocks can be checked by examining the contents of dual port RAM. After 8086-8089 protocol is established and 8089 software has been executed, the 8089 Prototype Board can be used for hardware development. Although the existing board can be used for basic hardware evaluation, there is plenty of area for design expansion with all the necessary interface signals available.

A block diagram of the 8089 Prototype Board design is shown in Figure 2. The design of the 8089 Prototype Board provides a fully arbitrated Multibus interface to an 8089 remote configuration. Since all program execution is done out of iSBC 86/12A dual port RAM, there is no local memory on the board. Positioned on the 8089 local bus providing RS232 serial communication to a CRT is an 8253 PIT (Programmable Interval Timer) and a 8251A USART (Universal Synchronous/Asynchronous Receiver/Transmitter). All the necessary decode and bus control logic is designed in the system. Inter-processor handshaking is accomplished with interrupts to the 86/12A and I/O-address-decoded channel attentions to the 8089.

Complete 8089 Prototype System operation can be best understood by reading the Demonstration Program Section covered later in this note. The following however, is a general overview of 8089 Prototype System operation. After both 8086 and 8089 programs are down-loaded into the iSBC 86/12A, the 8086 program can be entered. The 8086 program should first initialize any I/O or memory that may be used in conjunction with the 8089. For example, the

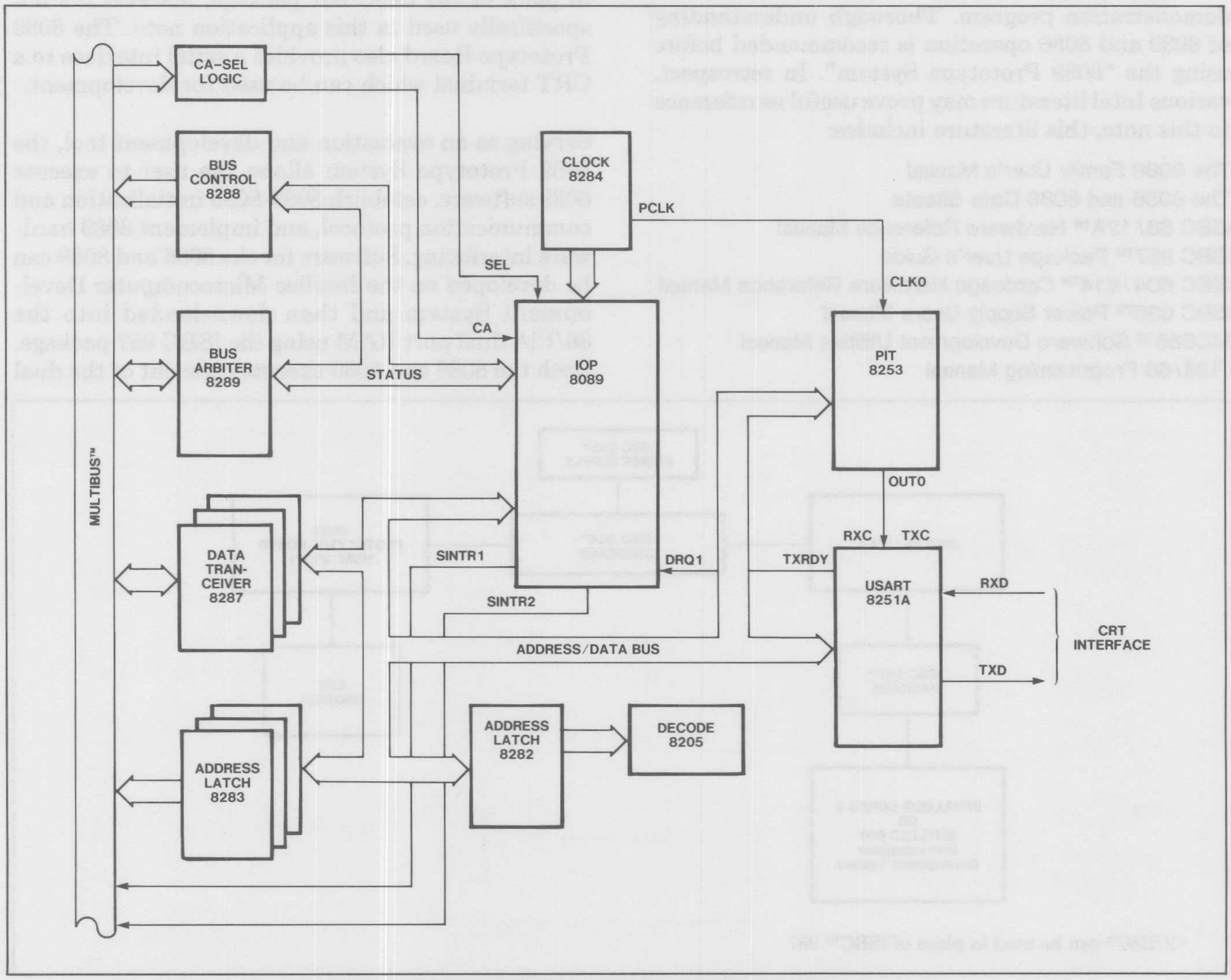


Figure 2. 8089 Prototype Board Block Diagram

AFN 01153A

8259A PIC (Programmable Interrupt Controller) and corresponding interrupt-vector-table memory locations fall into this category.

Before the 8089 can start its program execution the initialization and communication blocks must be set up by the 8086. To accomplish this, both the 8086 and the 8089 must be able to access the same memory locations, specifically upper memory (i.e. the 8089 system configuration pointer). On the 86/12A however, this memory area is reserved for the on board 957 Package Monitor ROM which is inaccessable from the system bus. To accommodate this situation the iSBC 86/12A dual port RAM, located at 0-7FFFH, is configured to allow a portion of it to appear as upper memory to the 8089 (FE000-FFFFFH). This same area is 6000-7FFFH to the 8086. Locations 0-5FFFFH aren't accessible to the 8089. Figure 3 shows a memory map of the iSBC 86/12A and 8089 memory.

This memory allocation allows the 8086 to set up the system configuration pointer at 8086 memory location 7FF6H, corresponding to 8089 memory location FFFF6H. The other initialization and communication blocks are also set up in this offset format. This includes the system configuration block, channel control block, parameter block, and task block. It should be noted that the prototype system isn't limited to using only the iSBC 86/12A dual port RAM, an external memory board may be used instead. If this method is used the iSBC 86/12A can be configured to address the same off board upper memory as the 8089 rather than the offset required when only using dual port RAM.

Once the 8089 initialization blocks are set up, the 8086 can send a CA (Channel Attention) to the 8089. The CA is I/O addressed at location 00H (SEL = 0) and 01H (SEL = 1). Upon the first CA, the 8089 will fetch the information from the system configuration pointer and the system configuration block for initialization. It then clears the busy flag in the channel control block. The SEL input has no effect on the 8089 during this first CA. The 8086 waits until the busy flag is cleared. Once this happens it then sets up the communication blocks and issues another CA, this time to initialize either channel 1 (SEL = 0) or channel 2 (SEL = 1). After receiving the CA the 8089 reads the appropriate channel control block. It then vectors to the task block designated by the task block pointer in the parameter block. This places the 8089 into program execution.

To exercise the 8089 Prototype Board an initial program should use peripheral devices provided for the serial communication, the 8253 and 8251A. This allows for a simple check-out procedure with visible results on a CRT terminal. The 8089 uses these devices in the demonstration program. Their address locations are shown in Table 1.

Table 1. 8089 Prototype Board I/O Addresses

Device	Function	8089 Address
8253	Counter 0	0E000H
	Counter 1	0E001H
	Counter 2	0E002H
	Mode	0E003H
8251A	Data	0C000H
	Command	0C001H

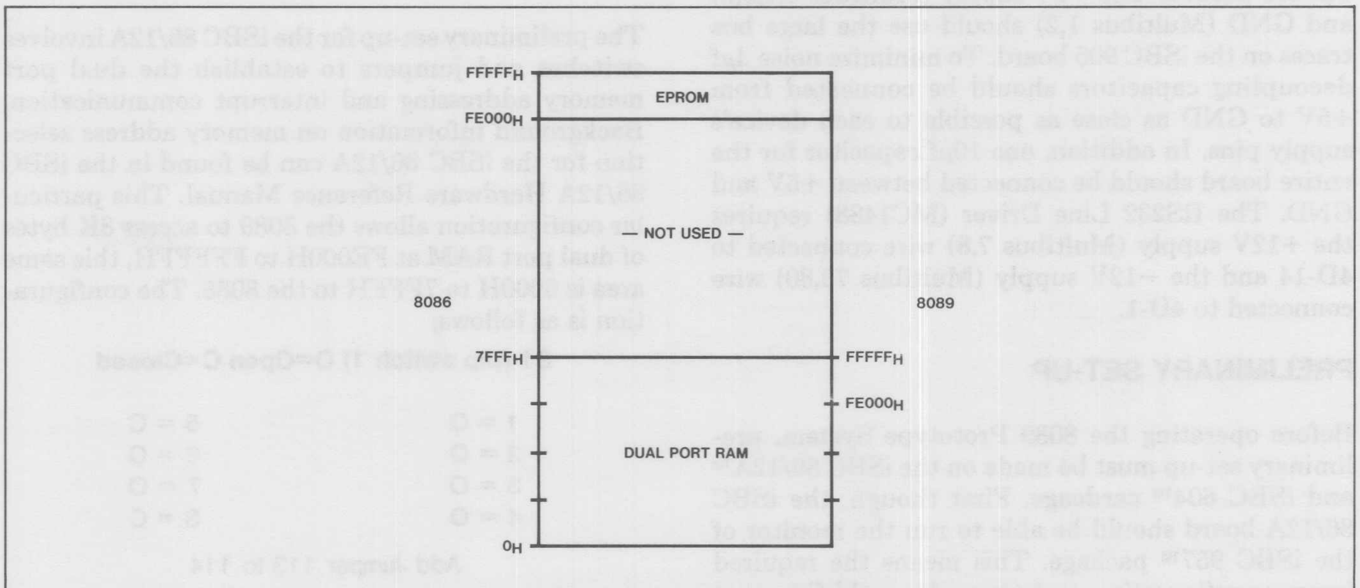


Figure 3. iSBC 86/12A and 8089 Memory Map

AFN 01153A

If the 8089 needs service from the 8086 during program execution, there are a number of ways to handle it. For instance, the 8089 SINTR1 or SINTR2 outputs can interrupt the 8086 through the 8259A. Another way might use handshaking flags in the parameter block that the 8086 polls to determine if the 8089 needs service. Additionally, halting 8089 program execution with the 8086 monitoring the busy flag can be used to indicate service is needed.

Since the dual port RAM contains the 8089 communication and task blocks, debugging 8089 code is possible. One method would set up break points in the task block. This can be done by replacing code at a certain location with a jump instruction. The breakpoint code would save the status of the 8089 and interrupt the 8086 for service. Break point operation can also be accomplished by inserting a HALT instruction. Another form of program check-out uses signal flags set or cleared throughout the program as a verification to proper execution. In the same manner, the 8089 CRT terminal interface can be useful by outputting to the CRT at various points throughout program execution.

8089 PROTOTYPE BOARD CONSTRUCTION

The following documentation is provided for construction of an 8089 Prototype Board to be used in an 8089 Prototype System. This includes a parts list (Table 2), components layout (Figure 4), and schematic (Figure 5). The complete design can be constructed on an iSBC 905™ Universal Prototype Board using wirewrap as a means for inter-connects. Note that power supply connections from Multibus™ to device are not shown in the schematic but are needed. The +5V supply (Multibus 3,4,5,6) and GND (Multibus 1,2) should use the large bus traces on the iSBC 905 board. To minimize noise .1μf decoupling capacitors should be connected from +5V to GND as close as possible to each device's supply pins. In addition, one 10μf capacitor for the entire board should be connected between +5V and GND. The RS232 Line Driver (MC1488) requires the +12V supply (Multibus 7,8) wire connected to 4D-14 and the −12V supply (Multibus 79,80) wire connected to 4D-1.

PRELIMINARY SET-UP

Before operating the 8089 Prototype System, preliminary set-up must be made on the iSBC 86/12A™ and iSBC 604™ cardcage. First though, the iSBC 86/12A board should be able to run the monitor of the iSBC 957™ package. This means the required jumper configuration, resistor packs, and I.C.'s must be added to conform with the directions in the iSBC

Table 2. 8089 Prototype Board Parts List

Quantity	Description
(NON IC'S) 1	SBC 905 Universal Prototype Board
1	40 pin wire wrap socket
1	28 pin wire wrap socket
1	24 pin wire wrap socket
9	20 pin wire wrap socket
1	18 pin wire wrap socket
2	16 pin wire wrap socket
8	14 pin wire wrap socket
1	15 MHz crystal
2	1K resistor
1	100K resistor
1	Normally off-momentary on switch
1	1 μF capacitor
23	.1 μF capacitor
1	10 μF capacitor
1	RS232 female connector
(ICS) 1	8089
1	8205
1	8251A
1	8253
1	8282
3	8283
1	8284
3	8287
1	8288
1	8289
1	8205
1	74S00
1	74S04
1	7427
1	74S32
1	74S74
1	74S126
1	74S133
1	MC1488
1	MC1489

957 package user's guide. Once monitor operation is verified, set-up for the 8089 Prototype System can begin.

The preliminary set-up for the iSBC 86/12A involves switches and jumpers to establish the dual port memory addressing and interrupt communication. Background information on memory address selection for the iSBC 86/12A can be found in the iSBC 86/12A Hardware Reference Manual. This particular configuration allows the 8089 to access 8K bytes of dual port RAM at FE000H to FFFFFH, this same area is 6000H to 7FFFH to the 8086. The configuration is as follows;

S1 (Dip switch 1) O=Open C=Closed

1 = O	5 = C
2 = O	6 = C
3 = O	7 = O
4 = O	8 = C

Add Jumper 113 to 114

Remove Jumpers 112 and 115 through 128

AFN 01153A

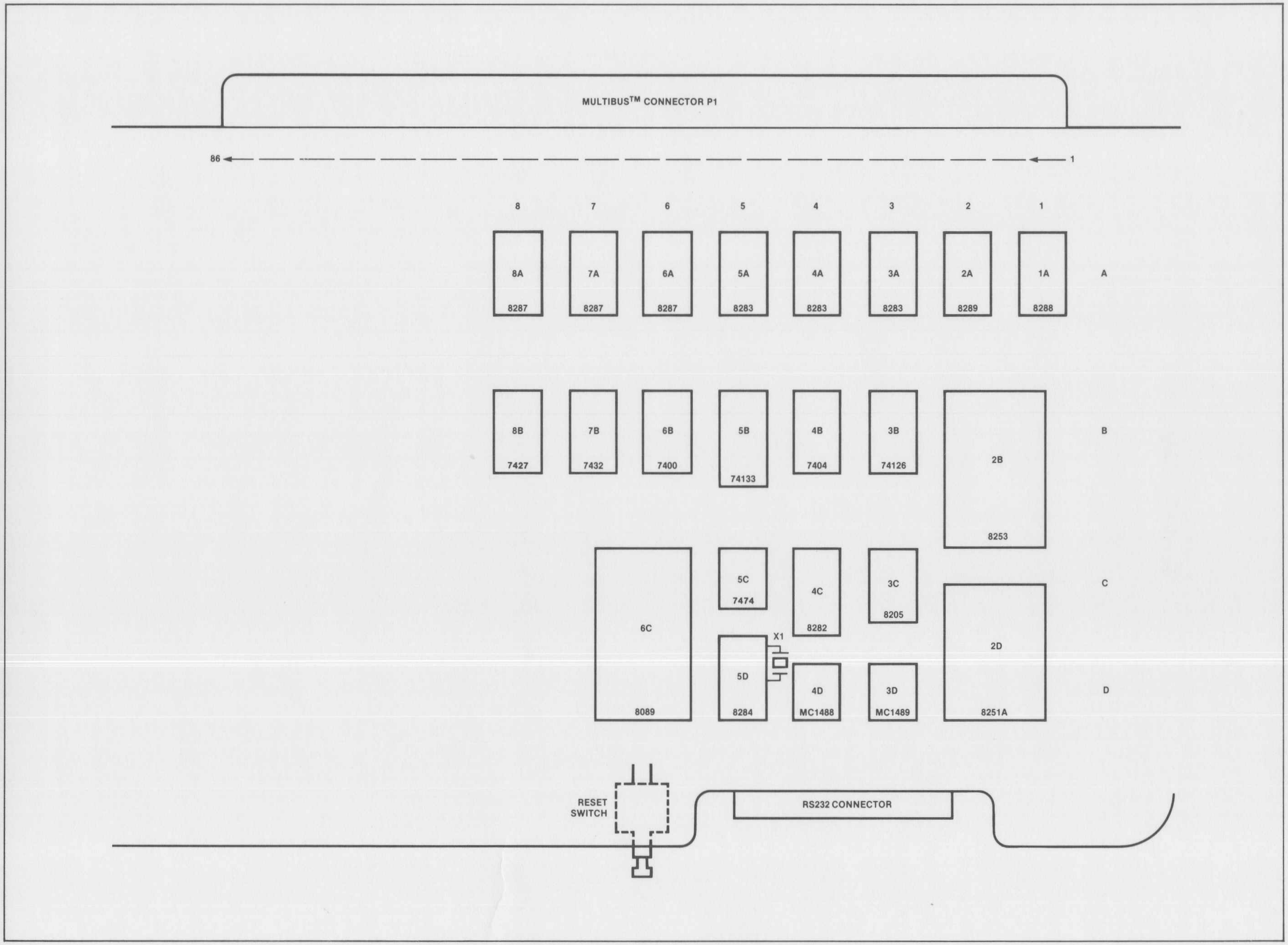
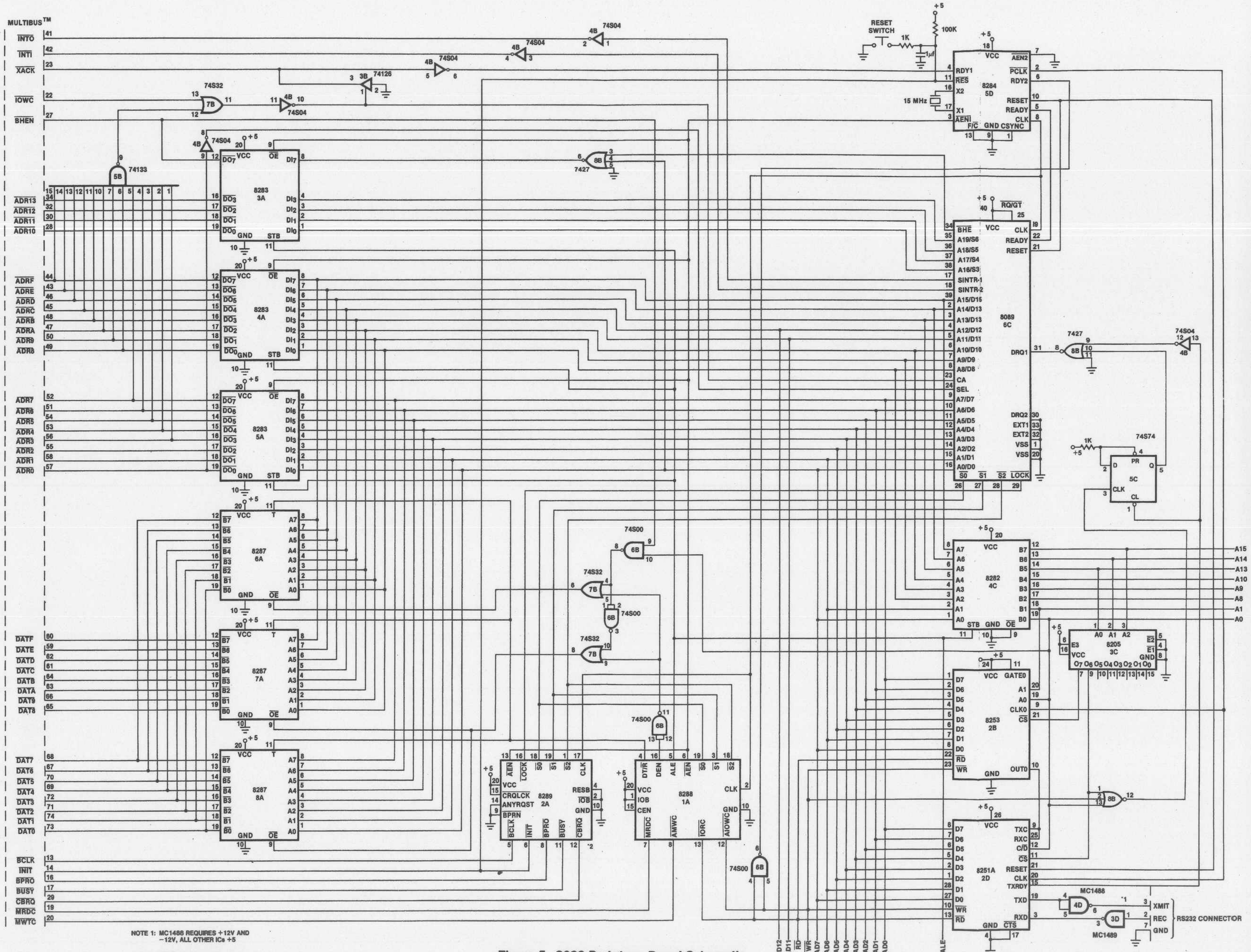


Figure 4. 8089 Prototype Board Components Layout

AFN 01153A



NOTE 1: MC1488 REQUIRES +12V AND −12V, ALL OTHER ICs +5

NOTE 2: CBRQ MUST BE TIED LOW IF ANOTHER BUS MASTER DOESN'T CONTROL IT.

Figure 5. 8089 Prototype Board Schematic

Interrupt matrix jumpers are added, allowing the INTR0 and INTR1 Multibus inputs driven by the 8089 to interrupt the 8086 via the 8259A IR0 and IR1. Note that when the 8259A is programmed all unused IR inputs should be masked.

Add Jumper 73 to 81

Add Jumper 72 to 80

The preliminary set-up for the iSBC 604 cardcage establishes Multibus bus priority. The 8089 Prototype Board should have highest priority and be located in the top slot of the cardcage (J2). The iSBC 86/12A should be in the bottom slot of the cardcage (J5). Bus priority is selected by connecting BPRO of the 8089 Prototype Board to BPRN of the iSBC 86/12A.

Add Jumper C to H

DEMONSTRATION PROGRAM

To demonstrate the operation of the 8089 Prototype System, a complete CPU-IOP program package, including PLM86 and ASM89 programs, are given. Together these two programs are good examples for 8086–8089 initialization and communication protocol. Not created to show the 8089 operation at its highest capabilities, programming was kept basic and tutorial, specifically for 8089 Prototype System demonstration and debug.

Program operation demonstrates the 8089 relieving the burden of the 8086 by handling message transfers to a CRT and processing message requests. Five messages and a menu are available for display on the CRT. The messages include information on various facets of the 8089 Prototype System. The menu shows the five different message titles that can be selected. To use the demonstration programs a few things should be noted. First, all the necessary changes in the Preliminary Set-Up section must be made. The programs must be compiled or assembled, linked and located, and then down-loaded or programmed into the iSBC 86/12A. This step is covered shortly in the Software Development Commands section. The CRT interfacing to the 8089 should be set at 9600 baud.

The PLM86 and ASM89 program listings are shown in Appendix A and B respectively. The name of the source programs are PROT89.SRC for the PLM86 program and 89DEMO.SRC for the ASM89 program. Both program listings are well documented with comments that help explain program operation.

System Program Flow

To understand the operation of the demonstration program an overview of the entire system program flow should first be examined. The system program flow can be broken into four sections, two for the PLM86 program and two for the ASM89 program. The PLM86 program consists of an initialization module and a message processing interrupt routine. The ASM89 program consists of an initialization task (Task 1) and a message processing task (Task 2). Figure 6 shows the overall system program flow and interprocessor handshaking of the demonstration programs.

Program flow starts with the PLM86 initialization module which is only executed once. Its basic functions are to set up the iSBC 86/12A memory and I/O and initialize the 8089 for program operation. The 8086 starts 8089 initialization by setting up the 8089 initialization blocks and issuing the first CA for 8089 internal initialization. The 8086 then monitors the busy flag in the Channel Control Block waiting for the 8089 to clear it. Once this occurs the 8086 sets up the 8089 communication blocks for Task 1 and issues a second CA to start Task 1. After all this is done the module stays in a tight loop waiting for an interrupt request from the 8089. This loop represents further 8086 program operation.

When the 8089 receives the second CA it starts executing Task 1. The primary function of Task 1 is to initialize the I/O on the 8089 local bus (i.e., 8251A and 8253) so it can support communication to the CRT terminal. It also sets up the parameter block to convey to the 8086 that the menu should be made available for display. The 8089 then issues an interrupt request to the 8086 via its SINTR1 output.

Upon interrupt the 8086 will enter its message processing routine. This routine examines the parameter block for a valid request from the 8089 and acts accordingly. As in the first case the 8089 Task 1 requested the menu, thus the 8086 will set up for the 8089 the menu and the necessary communication blocks to enter Task 2. The routine then issues a CA to the 8089 to start Task 2 program execution. Message requests from the 8089 are handled similar to that of menu request except that the routine decodes the request for selection of a particular message.

Once into Task 2 the 8089 will DMA transfer the menu or message to the CRT. Further requests from the CRT terminal are then monitored. If a valid request is made it's put into the parameter block for the 8086 and an interrupt request is sent to the 8086 to acquire the 8086's attention. From this point on

AFN 01153A

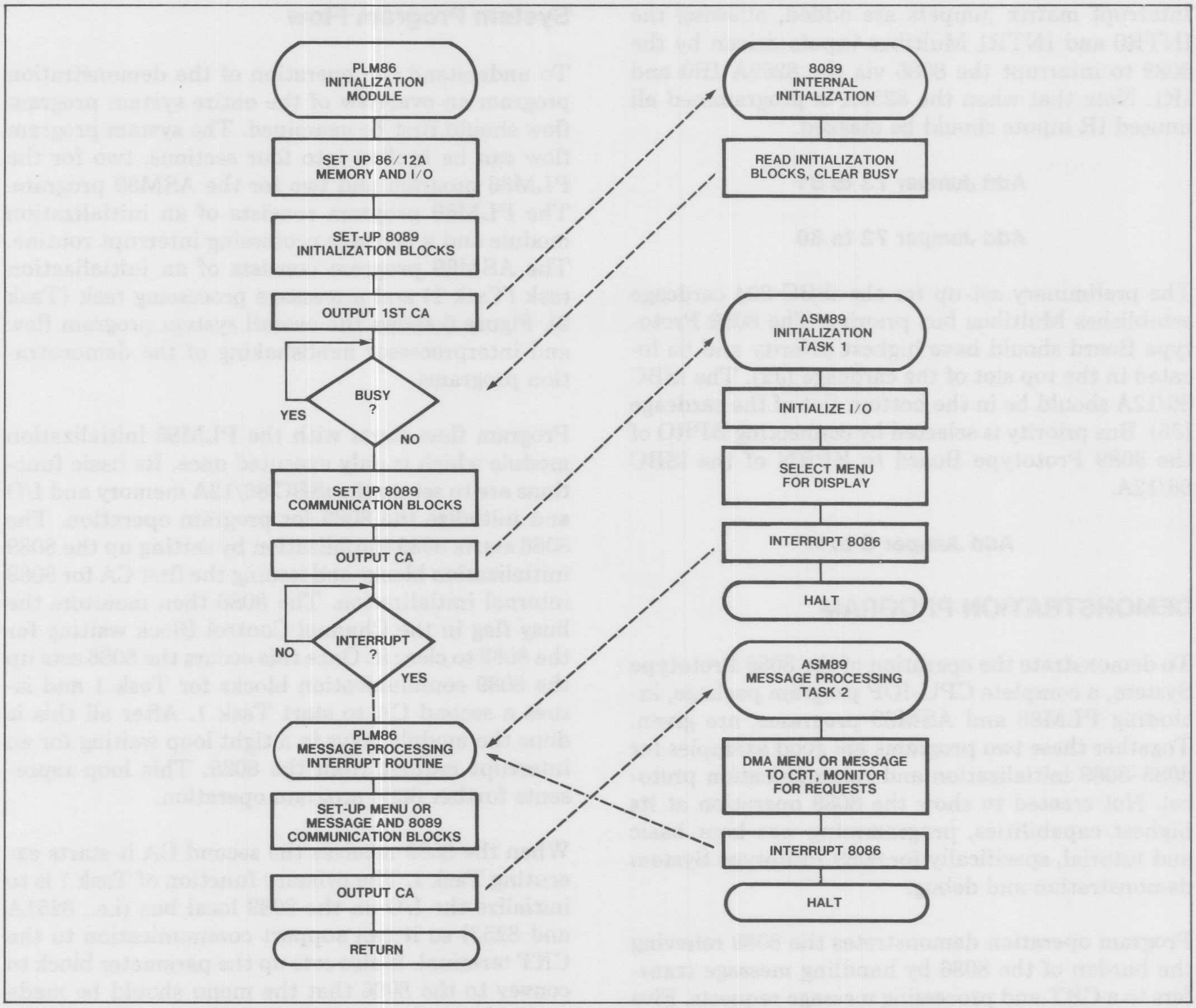


Figure 6. Overall System Program Flow for Demonstration Programs

system program flow is completely interrupt-CA driven between the 8086 and the 8089.

Figure 7 shows a memory map of the 8089 initialization and communication blocks set up by the 8086. The system configuration pointer, SYS BUS byte, and system configuration block fit the initialization blocks category. The channel control block, parameter block, and task block fit the communication blocks category. The initialization blocks are only used once after the first CA for 8089 internal initialization. The communication blocks are used throughout program execution.

The primary communication block for 8086–8089 interfacing is the parameter block. This block holds the necessary parameters used to communicate between the 8086 and 8089 throughout program execu-

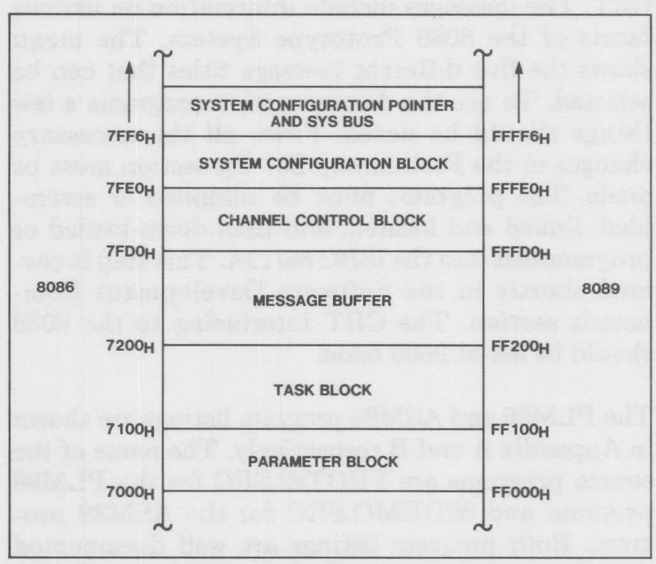


Figure 7. 8089 Initialization and Communication Block Memory Map

AFN 01153A

tion. Figure 8 shows the parameter block as it is used in the demonstration program. The first two words of the parameter block are dedicated for holding the task block address which vectors the 8089 to the task block after a CA. All other locations are defined for the particular needs of the demonstration program. In this case only three parameters are needed, the message buffer pointer, LEV byte and CI byte. The message buffer pointer is set up by the 8086 and used by the 8089. It references to the message buffer when transferring the menu or a message to the CRT. The LEV byte (Level) is set by the 8086 and used by the 8089 to determine proper decode for CRT terminal requests. The CI byte (Console In) is used to store a valid request from the 8089 monitored CRT terminal to the 8086.

Now with an overview of system program flow and background information on initialization and communication blocks, let's delve into the more detailed aspects of post initialization program flow. Figure 9 displays the program flow for the PLM86 message processing interrupt routine. The first thing done in the routine is the examination of the CI byte in the parameter block. The contents of the CI byte defines whether the menu or a message should be set up for the 8089. If CI contains an ASCII "Y" (for yes), the menu will be moved to the message buffer and the LEV byte in the parameter block will be set false. If CI doesn't contain "Y", it will contain a message number "1–5" that is decoded to select a particular message to be moved to the message buffer. The LEV byte is then set true. The function of the LEV byte is to convey to the 8089 whether the 8086 has provided a menu or message in the message buffer. In the first occurrence of this interrupt routine the 8089 Task 1 sets CI to "Y", thus the menu is moved to the message buffer and LEV is set false.

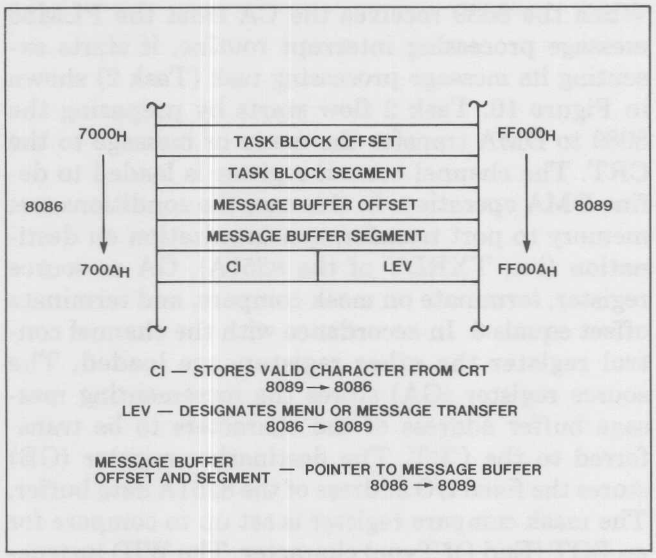


Figure 8. Parameter Block Designation for Demonstration Programs

After the menu or message has been prepared for the 8089, the PLM86 interrupt routine sets up the communication blocks for the 8089 Task 2. First the code for the task block is set up and the task block address is written into the parameter block. Then the channel control block is set to define 8089 operation and to reference the parameter block. A CA is issued to the 8089 to start it executing Task 2.

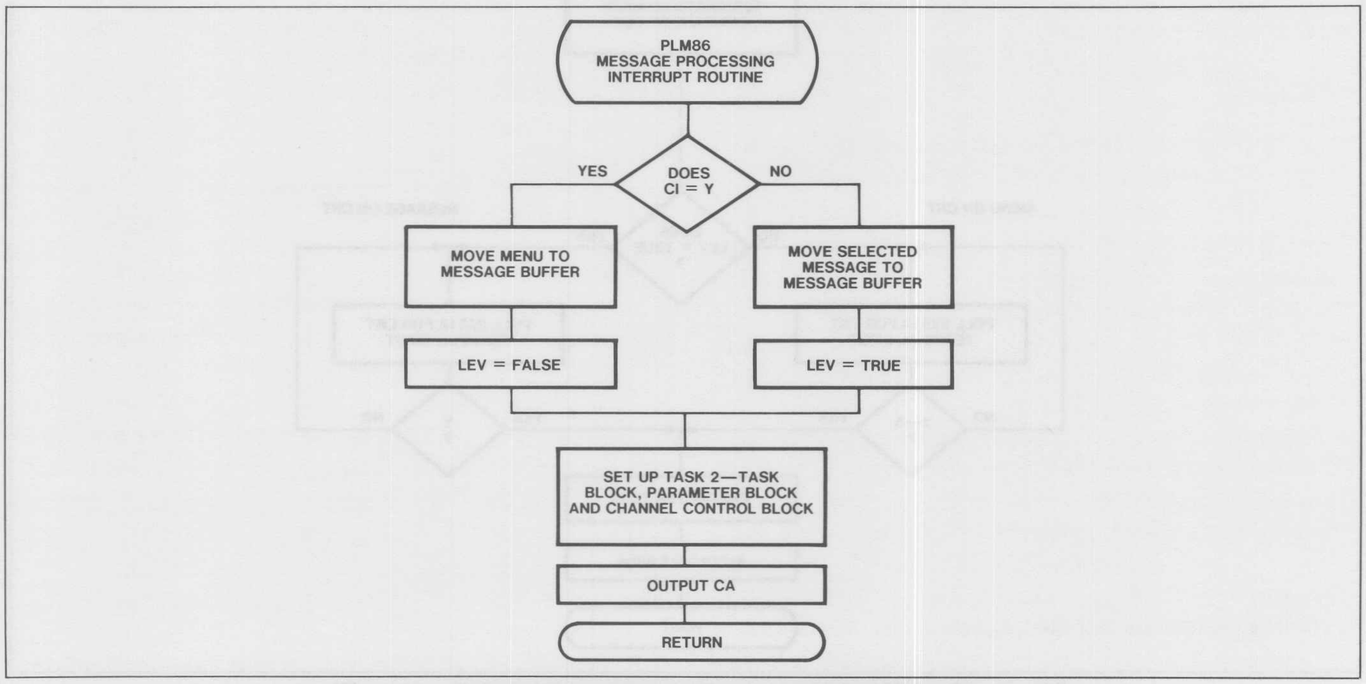


Figure 9. PLM86 Message Processing Interrupt Routine Program Flow

AFN 01153A

When the 8089 receives the CA from the PLM86 message processing interrupt routine, it starts executing its message processing task (Task 2) shown in Figure 10. Task 2 flow starts by preparing the 8089 to DMA transfer the menu or message to the CRT. The channel control register is loaded to define DMA operation. In this case the conditions are: memory to port transfer, synchronization on destination (i.e., TXRDY of the 8251A), GA as source register, terminate on mask compare, and terminate offset equals 0. In accordance with the channel control register the other registers are loaded. The source register (GA) stores the incrementing message buffer address of the characters to be transferred to the CRT. The destination register (GB) stores the fixed I/O address of the 8251A data buffer. The mask compare register is set up to compare for an EOT (End Of Type) character. The WID instruction is then executed to define the bus configuration for the DMA, in this case 16 bit to 8 bit.

After all the registers are initialized and the WID instruction has been executed the XFER instruction initiates the DMA cycle. At this point whenever the 8251A TXRDY causes DRQ1 to go high, DMA will occur. Each transfer is checked via the mask compare register for an EOT character. When the EOT character is found DMA will terminate leaving the completed message displayed on the CRT. Task block execution resumes at the next instruction after the XFER instruction.

Once the complete message is transferred the 8089 is ready to monitor the CRT terminal for message requests. In order to know what requests are valid the LEV byte must be interrogated to determine whether the menu or message had just been transferred. If LEV is true then a message was just transferred and the CRT terminal input is polled for a "Y" character. The "Y" character can be entered at the end of each message to cause the menu to be displayed allowing further message selection. If the LEV byte is false then the menu has just been transferred and the CRT keyboard is polled for character "1–5". The "1–5" characters can be entered at the end of the menu to select a particular message to be displayed. The CRT terminal will continue to be monitored until a valid request is encountered.

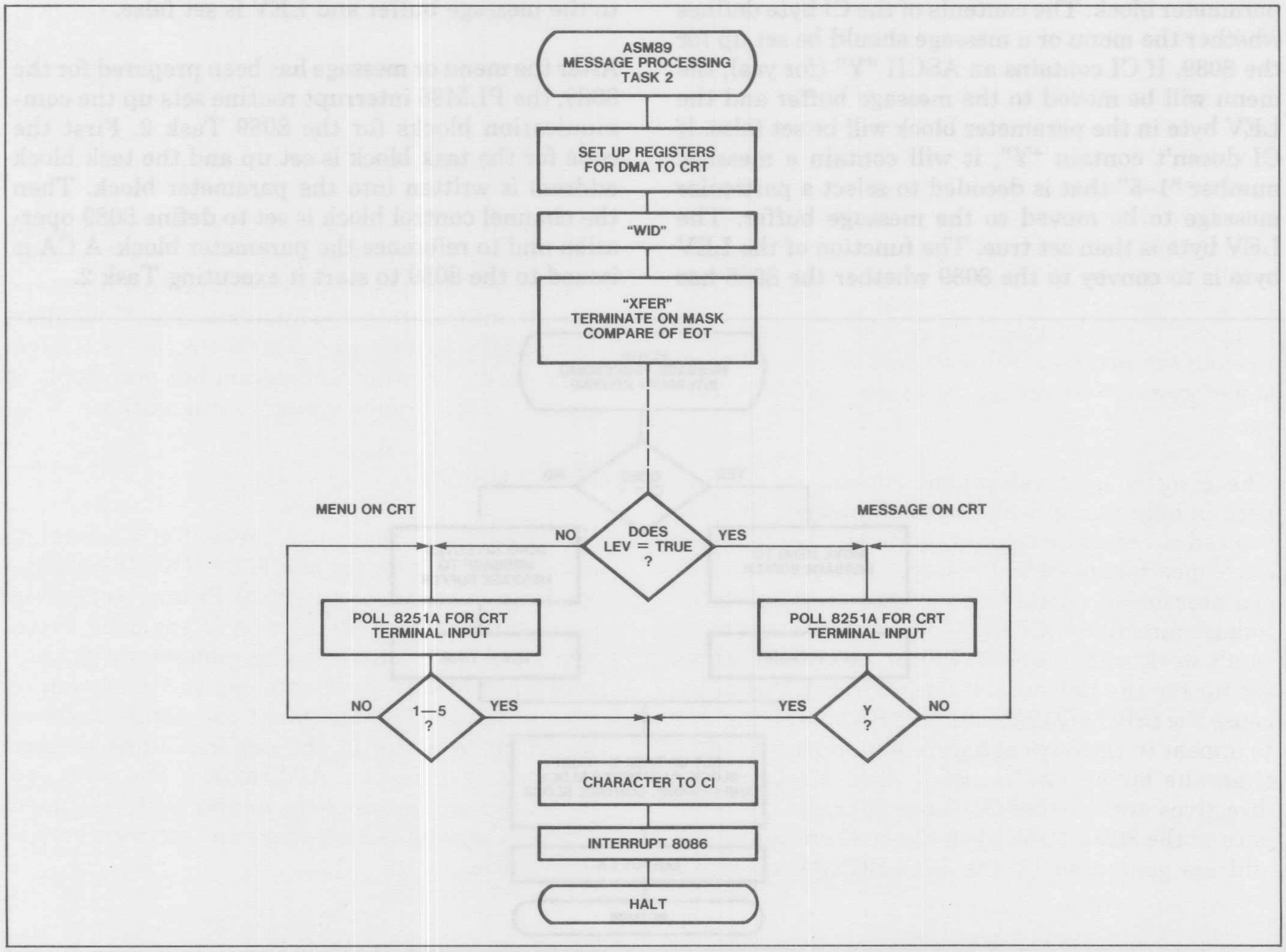


Figure 10. ASM89 Message Processing, Task 2 Program Flow

AFN 01153A

When this happens the request is moved into the CI byte in the parameter block. The 8089 then interrupts the 8086 and halts. In response, the 8086 goes back into its message processing interrupt routine. Henceforth, the entire cycle is repeated.

Note that for this particular program it's only necessary to set up the communication blocks once rather than repeating the process during every 8086 interrupt routine. It was done this way however, to demonstrate an alternative method of setting up the communication blocks and alleviate possible problems encountered when debugging the 8089 Prototype System. Since this method restores the state of the dual port RAM after each interrupt, the possibility of memory getting destroyed by faulty hardware or software is minimized. This also applies to the way the menu or message was set up for the 8089. Rather than reloading the message buffer each interrupt routine, linkage to the different messages could be accomplished by initially loading all messages in system RAM and merely changing the message buffer pointer in the parameter block.

Software Development Commands

This section shows the software development commands used to generate the demonstration program hex code for the iSBC 86/12A (Figure 11). Two methods are shown, one for down-loading and one for PROM programming. The down-loading method locates the program in iSBC 86/12A dual port RAM at 200H by LOC86 default. The PROM programming method relocates the program to FE000H, the location of the PROM area on the iSBC 86/12A. Placing the program into PROM makes it a "stand alone" system without the use of the iSBC 957 package.

These software development commands needn't pertain only to the demonstration program but can be used as reference for other software that might be developed for the 8089 Prototype System. One special precaution should be noted, use of the program linkage directive EXTERNAL in an 8089 task block won't work with the 8089 Prototype System as it's set up for the demonstration program. This is because the iSBC 86/12A dual port RAM is configured to appear to the 8089 at a different address location than the 8086. The PUBLIC and EXTERNAL directives are assigned absolute addresses in reference to the 8086. Thus when the 8089 encounters an address generated by the EXTERNAL directive, the address won't have the necessary offset to account for the dual port RAM configuration. To use the directives, addressing must be common. An alternative approach to using the directives can be accomplished by setting up pointers or variables referenced in the parameter block. However, if this doesn't suffice and using the linkage directives is still necessary, off board memory with common addressing must be used rather than the dual port RAM offset method. PUBLIC and EXTERNAL directives are used in the demonstration program to reference the 8089 task blocks for relocation in dual port RAM. The use of the directives however, is in respect to 8086 absolute addressing.

8089 SYSTEM DEBUG

There's always the "slight possibility" that the first attempt to use the 8089 Prototype System may prove to be a failure. Don't lose hope, this is a very normal occurrence working with any wire wrap constructed prototype board, especially when preliminary hardware and software set-up is involved. First, review the Preliminary Set-up section making sure all jumpers and switches are correctly positioned on the iSBC 86/12A™ and iSBC 604™ Cardcage and that the iSBC 86/12A can operate alone. Then, verify the software development for the 8086 and 8089 programs. If these conditions are found proper then a system debug effort must be undertaken.

To simplify the debug process an 8089 system debug flow chart is provided in Figure 12. This flow chart should help pin point where the problem lies and provide information for analysis. The flow chart needn't only pertain to the 8089 Prototype System using the demonstration program but may apply to any 8086–8089 remote system configuration.

CONCLUSION

This application note has presented a complete prototyping system for the 8089 IOP. It has provided: an overview of the 8089 Prototype System, documentation for construction of the 8089 Prototype Board, explanation of a demonstration program, its software development, and a section on system debug. The combination of this material should prove useful in the understanding of basic 8089 system operation. Additionally, the 8089 Prototype System should help minimize development efforts by serving as a hardware and software evaluation vehicle.

Down-Loading Method—default starting address 200H

```
PLM86   PROT89.SRC

ASM89   89DEMO.SRC

LINK86  PROT89.OBJ, 89DEMO.OBJ TO PROLNK.OBJ

LOC86   PROLNK.OBJ TO PROLOC.OBJ

OH86   PROLOC.OBJ TO PROLOC.HEX
```

PROM Method—starting address FE000H

```
PLM86   PROT89.SRC

ASM89   89DEMO.SRC

LINK86  PROT89.OBJ, 89DEMO.OBJ TO PROLNK.OBJ

LOC86   PROLNK.OBJ TO PROPRM.OBJ&
BOOTSTRAP&
ADDRESSES (SEGMENTS(PROTOTYPE89.CODE(FE000H),DEMO(FFB00H)))

OH86   PROPRM.OBJ TO PROPRM.HEX
```

UPP programming, four 2716's

```
      READ 86HEX FILE PROPRM.HEX INTO 2000H

      STRIP LOW FROM 0 TO 1FFFH INTO 4000H

      STRIP HIGH FROM 0 TO 1FFFH INTO 6000H

(1st) PROGRAM FROM 4000H TO 47FFH START 0

(2nd) PROGRAM FROM 4800H TO 4FFFH START 0

(3rd) PROGRAM FROM 6000H TO 67FFH START 0

(4th) PROGRAM FROM 6800H TO 6FFFH START 0
```

Socket Insertion

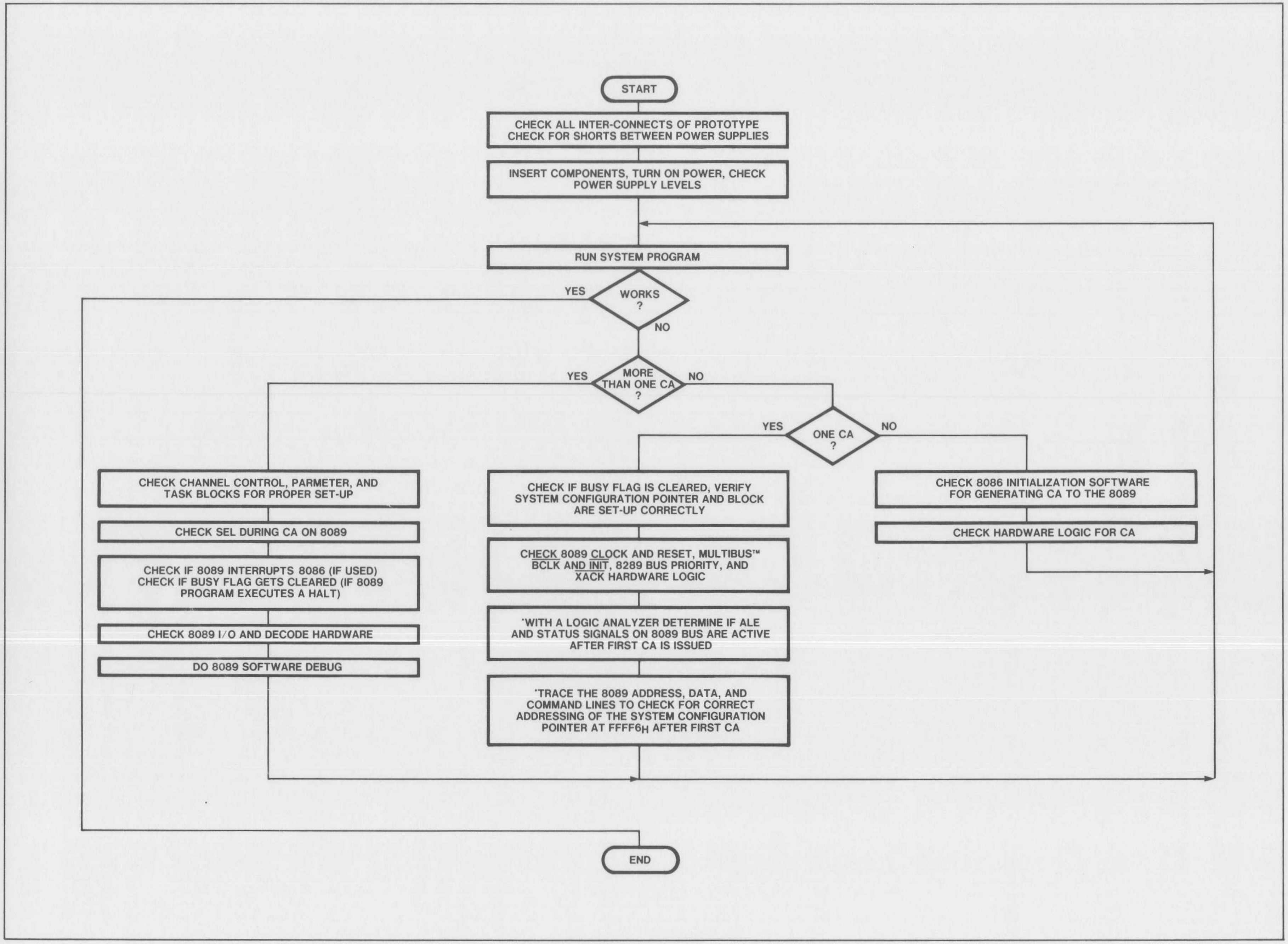
2716#	86/12A Socket
1	A28
2	A29
3	A46
4	A47

Figure 11. Software Development Commands

The demonstration program source files are available on floppy disk or paper tape through Insite™, Intels User's Library. For information about Insite, write to:

Insite™
Intel Corp.
3065 Bowers Avenue
Santa Clara, CA 95051

AFN 01153A



AFN 01153A

Figure 12. 8089 System Debug Flow Chart

*NOTE: THIS CAN BE ACCOMPLISHED BY USING TWO SYNCHRONIZED PULSE GENERATORS TO REPEATEDLY PULSE RESET THEN CA (THESE PINS SHOULD ONLY BE CONNECTED TO THE PULSE GENERATORS).

Appendix A

Appendix A

```
PL/M-86 COMPILER    8089 PROTOTYPE DEMO                                   PAGE   1


ISIS-II PL/M-86 V1.1 COMPILATION OF MODULE PROTOTYPE89
OBJECT MODULE PLACED IN :F1:PROT89.OBJ
COMPILER INVOKED BY:   PLM86 :F1:PROT89.SRC


              $TITLE('8089 PROTOTYPE DEMO') LARGE OPTIMIZE(2)
   1          PROTOTYPE$89:   DO;
              /***************************************************************************/
              /*                                                                         */
              /*          PLM86 DEMONSTRATION PROGRAM FOR THE 8089 PROTOTYPE SYSTEM      */
              /*                                                                         */
              /***************************************************************************/

              /***************************************************************************/
              /*                                                                         */
              /*                          LITERAL DECLARATIONS                           */
              /*                                                                         */
              /***************************************************************************/

              /* 8259A LITERALS */

   2   1          DECLARE

                      INT$STAT$PORT       LITERALLY       '0C0H',
                      INT$MASK$PORT       LITERALLY       '0C2H',
                      INT$ICW1            LITERALLY       '13H' ,
                      INT$ICW2            LITERALLY       '50H' ,
                      INT$ICW4            LITERALLY       '0FH' ,
                      INT$MASK            LITERALLY       '0FEH';


              /*  RAM LOCATIONS FOR THE 8086 */

   3   1          DECLARE

                      SINT$BASE   LITERALLY     '7FF6H' , /* SYSTEM INITALIZATION BLOCK */
                      SCB$BASE    LITERALLY     '7FE0H' , /* SYSTEM CONTROL BLOCK */
                      CB$BASE     LITERALLY     '7FD0H' , /* COMMAND BLOCK */
                      PB$BASE     LITERALLY     '7000H' , /* PARAMETER BLOCK */
                      TB$BASE     LITERALLY     '7100H' , /* TASK BLOCK */
                      MSG$BASE    LITERALLY     '7200H' , /* DISPLAY MESSAGE BUFFER */
                      INTR$TYPE   LITERALLY     '0140H' ; /* INTERRUPT VECTOR TABLE */


              /*   RAM LOCATIONS FOR THE 8089 */

   4   1          DECLARE

                      SCB$89      LITERALLY     '0FFFE0H' , /* SYSTEM CONTROL BLOCK */
                      CB$89       LITERALLY     '0FFFD0H' , /* COMMAND BLOCK */
                      PB$89       LITERALLY     '0FF000H' , /* PARAMETER BLOCK */
                      TB$89       LITERALLY     '0FF100H' , /* TASK BLOCK */
                      MSG$89      LITERALLY     '0FF200H' ; /* DISPLAY MESSAGE BUFFER */
```

AFN 01153A

Appendix A (Continued)

```
PL/M-86 COMPILER    8089 PROTOTYPE DEMO                                                PAGE   2


            /*     8089 CCW'S */

   5   1          DECLARE

                       INIT$CCW          LITERALLY     '13H' , /* I/O INITALIZATION CCW */
                                                               /* ENABLE INTERRUPTS     */
                                                               /* EXECUTE TASK BLOCK IN */
                                                               /* SYSTEM MEMORY         */
                       DSP$CCW           LITERALLY     '0BH' ; /* DISPLAY MESSAGE CCW */
                                                               /* RESET INTERRUPT       */
                                                               /* EXECUTE TASK BLOCK  */
                                                               /* IN SYSTEM MEMORY     */


            /*     8089 INITIALIZATION COMMANDS */

   6   1          DECLARE

                       SOC$CMD           LITERALLY     '00H' , /* 8 BIT I/O BUS */
                       SYSBUS$CMD        LITERALLY     '01H' ; /* 16 BIT SYSTEM BUS */


            /*     8089 CHANNEL ATTENTION  */

   7   1          DECLARE

                       CHAN$ATT          LITERALLY     '00H' ;


            /*  MISCELLANEOUS DECLARATIONS  */

   8   1          DECLARE

                       BUSYSTATUS    LITERALLY      '0FFH',
                       TRUE          LITERALLY      '0FFH',
                       FALSE         LITERALLY      '00H',
                       NMBR$MSK      LITERALLY      '07H',
                       CR            LITERALLY      '0DH',
                       LF            LITERALLY      '0AH',
                       ESC           LITERALLY      '1BH',
                       E             LITERALLY      '45H',
                       EOT           LITERALLY      '04H';
```

Appendix A (Continued)

```
PL/M-86 COMPILER    8089 PROTOTYPE DEMO                                    PAGE  3


                  /*************************************************************/
                  /*                                                           */
                  /*                    RAM DECLARATIONS                       */
                  /*                                                           */
                  /*************************************************************/

   9   1      DECLARE     SINT    STRUCTURE (SYSBUS WORD, SCB$PTR POINTER) AT (SINT$BASE);

                          /*******************************************/
                          /*                  \    SYSBUS COMMAND    */
                          /*******************************************/
                          /*          SCB         OFFSET             */
                          /*******************************************/
                          /*          SCB         SEGMENT            */
                          /*******************************************/

  10   1      DECLARE     SCB    STRUCTURE (SOC WORD, CB$PTR POINTER) AT (SCB$BASE);

                           /*********************************************/
                           /*                 \   SOC  COMMAND          */
                           /*********************************************/
                           /*      COMMAND  BLOCK  OFFSET               */
                           /*********************************************/
                           /*      COMMAND  BLOCK  SEGMENT              */
                           /*********************************************/


  11   1      DECLARE     CB(2) STRUCTURE (CCW BYTE, BUSY BYTE, PB$PTR POINTER,
                                DUMMY WORD) AT (CB$BASE);

                           /*********************************************/
                           /*    BUSY FLAG        \         CCW         */
                           /*********************************************/
                           /*        PARAMETER  BLOCK  OFFSET           */
                           /*********************************************/
                           /*        PARAMETER  BLOCK  SEGMENT          */
                           /*********************************************/
                           /*                DUMMY    WORD              */
                           /*********************************************/

                  /*        THE ABOVE COMMAND BLOCK FORMAT IS THE STRUCTURE FORMAT     */
                  /*        THE CB ARRAY CONTAINS TWO STRUCTURES; ONE FOR EACH         */
                  /*        CHANNEL OF THE 8089.                                       */


  12   1      DECLARE     PB     STRUCTURE (TB$PTR POINTER, MSG$PTR POINTER,
                                         LEVEL BYTE, CI BYTE) AT (PB$BASE);
```

AFN 01153A

Appendix A (Continued)


```
                    /*****************************************************/
                    /*            TASK   BLOCK   OFFSET                  */
                    /*****************************************************/
                    /*            TASK   BLOCK   SEGMENT                 */
                    /*****************************************************/
                    /*        MESSAGE  BUFFER  OFFSET                    */
                    /*****************************************************/
                    /*        MESSAGE  BUFFER  SEGMENT                   */
                    /*****************************************************/
                    /*  CHARACTER FROM CRT  \ DISPLAY LEVEL CMD TO IOP   */
                    /*****************************************************/


 13   1       DECLARE      TB      (512) BYTE AT (TB$BASE);

                    /*****************************************************/
                    /*     RAM BUFFER FOR TASK BLOCK PROGRAM             */
                    /*****************************************************/

 14   1       DECLARE      MSG$BUF (1024) BYTE AT (MSG$BASE);

                     /****************************************************/
                     /*        DISPLAY     MESSAGE      BUFFER           */
                     /****************************************************/


 15   1       DECLARE      INTR$VEC$80  POINTER AT (INTR$TYPE);

 16   1       DECLARE      INTR$IP$80   WORD    AT (INTR$TYPE);


          /*************************************************************************/
          /*                                                                       */
          /*                  ROM DECLARATION AND INITIALIZATION                   */
          /*                                                                       */
          /*************************************************************************/

 17   1           DECLARE      MENU(*) BYTE  DATA (CR,LF,ESC,E,
          '                  ***********************************',CR,LF,
          '                  *                                 *',CR,LF,
          '                  *     THE 8089 PROTOTYPE SYSTEM   *',CR,LF,
          '                  *      DEMONSTRATION PROGRAM      *',CR,LF,
          '                  *                                 *',CR,LF,
          '                  ***********************************',CR,LF,
          CR,LF,LF,
```

AFN 01153A

Appendix A (Continued)


```
              '                    SELECTION                TOPIC',CR,LF,LF,
              '                        1          WHAT IS THE 8089 IOP ?',CR,LF,LF,
              '                        2          WHAT IS THE 8289 BUS ARBITER ?',CR,LF,LF,
              '                        3          ABOUT THIS DEMONSTRATION',CR,LF,LF,
              '                        4          8089 INITALIZATION PROTOCOL',CR,LF,LF,
              '                        5          8089 COMMUNICATION PROTOCOL',CR,LF,LF,
              LF,LF,
              '            FOR ADDITIONAL INFORMATION ON THE ABOVE TOPICS',CR,LF,
              '            PLEASE SELECT THE APPROPRIATE ENTRY (1,2,3,4,5) - ',EOT,EOT);

 18   1       DECLARE   MSG1(*) BYTE DATA(CR,LF,ESC,E,
              '                  8089 I/O PROCESSOR',
              CR,LF,LF,LF,
              '    THE 8089 I/O PROCESSOR IS A FIRST OF ITS KIND SYSTEMS COMPONENT.  IT',
              CR,LF,LF,
              'USES THE CONCEPT OF A CHANNEL CONTROLLER, COMMON IN MAINFRAMES, TO SOLVE',
              CR,LF,LF,
              'THE I/O PROCESSING AND HIGH PERFORMANCE DMA REQUIREMENTS OF MICROPROCESSOR',
              CR,LF,LF,
              'SYSTEMS.  THE 8089 CAN BE USED IN CONJUNCTION WITH THE 8086 (16 BIT BUS)',
              CR,LF,LF,
              'OR 8088 (8 BIT BUS) AND 8 OR 16 BIT PERIPHERALS TO SIGNIFICANTLY ENHANCE',
              CR,LF,LF,
              'SYSTEM PERFORMANCE.  THE 8089 OFFLOADS I/O FROM THE HOST CPU AND PROCESSES',
              CR,LF,LF,
              'CONCURRENTLY WITH CPU ACTIVITY.   ALSO, THE 8089 ADDS INTELLIGENCE TO THE',
              CR,LF,LF,
              'PERIPHERAL SUBSYSTEM WHILE MODULARIZING AND SIMPLIFING THE SYSTEM I/O. ',
              CR,LF,LF,
              'EACH IOP HAS TWO I/O CHANNELS THAT CAN PROVIDE DMA AT 1.25 MEGABYTE/SEC,',
              CR,LF,LF,
              'PROCESS INDEPENDENT PROGRAMS, AND HANDLE MULTIPLE I/O DEVICES. ',
              CR,LF,LF,
                '              TO SELECT ANOTHER MESSAGE TYPE Y-',EOT,EOT);

 19   1       DECLARE    MSG2(*) BYTE DATA (CR,LF,ESC,E,
              '                   THE 8289 BUS ARBITER',
              CR,LF,LF,LF,
              '      THE 8289 BUS ARBITER PROVIDES THE HARDWARE MECHANISMS FOR INTER-',
              CR,LF,LF,
              'PROCESSOR COMMUNICATION AND SHARED RESOURCES IN A MULTIPLE CPU SYSTEM.  THE',
              CR,LF,LF,
              '8289 FEATURES SEVERAL USER DEFINABLE PRIORITIZATION AND BUS CONFIGURATIONS. ',
              CR,LF,LF,
              'DEMONSTRATED HERE, THE RESB MODE SEPERATES 86/12 PRIVATE RESOURCES FROM',
              CR,LF,LF,
              'SYSTEM BUS SHARED RESOURCES, WHILE THE IOB MODE DIVIDES THE 8089 I/O BUS',
              CR,LF,LF,
              'FROM THE SYSTEM BUS.  IN BOTH CASES THE 8289 COMPLETELY ARBITRATES SYSTEM',
              CR,LF,LF,
              'BUS USAGE TO MANAGE MULTIPLE PROCESSOR CONTENTION. ',
              CR,LF,LF,
              '      THE 8086 FAMILY AND MULTIBUS CONCEPT ALLOWS PARTITIONING APPLICATIONS',
              CR,LF,LF,
              'INTO SMALLER MORE MANAGEABLE TASKS.  THUS, ADDING NEW FUNCTIONS OR UPGRADING',
```

AFN 01153A

Appendix A (Continued)


```
              CR,LF,LF,
              'EXISTING ONES WILL HAVE MINIMAL EFFECT ON THE ORIGINAL DESIGN.',
              CR,LF,LF,
              '                    TO SELECT ANOTHER MESSAGE TYPE Y-',EOT);

20   1        DECLARE      MSG3(*) BYTE  DATA(CR,LF,ESC,E,
                '                       ABOUT THIS DEMONSTRATION',
              CR,LF,LF,LF,
              '    TO DEMONSTRATE THE 8086 FAMILY CPU-IOP CONCEPT, AN SBC 86/12 AND AN 8089',
              CR,LF,LF,
              'PROTOTYPE BOARD ARE INTERFACED VIA THE INTEL MULTIBUS. IN THIS DEMO THE 8089',
              CR,LF,LF,
              'UNBURDENS THE 8086 BY HANDLING MESSAGE TRANSFERS TO THE CRT AND PROCESSING',
              CR,LF,LF,
              'MESSAGE REQUESTS. OPERATION IS AS FOLLOWS: USING A CHANNEL ATTENTION (CA) THE',
              CR,LF,LF,
              '8086 INITIALIZES THE 8089 AND CAUSES IT TO EXECUTE A TASK BLOCK TO PROGRAM',
              CR,LF,LF,
              'THE PERIPHERAL DEVICES ON ITS LOCAL BUS. THE 8089 THEN INTERRUPTS THE 8086',
              CR,LF,LF,
              'TO REQUEST A MESSAGE FOR DISPLAY. RESPONDING, THE 8086 SETS UP LINKAGE TO',
              CR,LF,LF,
              'THE TASK BLOCK PROGRAM AND ISSUES A CA TO THE 8089. AFTER EACH CA THE 8089',
              CR,LF,LF,
              'DISPLAYS THE MESSAGE, POLLS THE CRT TERMINAL FOR A VALID MESSAGE REQUEST AND',
              CR,LF,LF,
              'THEN INTERRUPTS THE 8086. HENCEFORTH THE CYCLE IS REPEATED.',
              CR,LF,LF,
                '                     TO SELECT ANOTHER MESSAGE TYPE Y-',EOT,EOT);
21   1        DECLARE     MSG4(*) BYTE       DATA (CR,LF,ESC,E,
              '                      8089  INITIALIZATION PROTOCAL',
              CR,LF,LF,LF,
              'SYSTEM INITALIZATION        ++++++++++++++++++++++++++++++++++++++++++++++',
              CR,LF,
              '                            +                     +  SYSBUS COMMAND     +',
              CR,LF,
              '                            ++++++++++++++++++++++++++++++++++++++++++++++',
              CR,LF,
              '                            +        SYSTEM CONTROL BLOCK ADDRESS       +',
              CR,LF,
              '                            ++++++++++++++++++++++++++++++++++++++++++++++',
              CR,LF,
              'SYSTEM CONTROL BLOCK        ++++++++++++++++++++++++++++++++++++++++++++++',
              CR,LF,
              '                            +                     +  SOC  COMMAND       +',
              CR,LF,
              '                            ++++++++++++++++++++++++++++++++++++++++++++++',
              CR,LF,
              '                            :            COMMAND  BLOCK  ADDRESS        +',
              CR,LF,
              '                            ++++++++++++++++++++++++++++++++++++++++++++++',
              CR,LF,LF,LF,
              '       ON THE FIRST CHANNEL ATTENTION AFTER RESET, THE IOP READS THESE',
              CR,LF,LF,
```

AFN01153A

Appendix A (Continued)

PL/M-86 COMPILER   8089 PROTOTYPE DEMO

```
         'CONTROL BLOCKS TO DETERMINE THE WIDTH OF THE SYSTEM BUS (8 OR 16), THE',
         CR,LF,LF,
         'I/O BUS WIDTH (8 OR 16), PRIORITY INFORMATION, AND WHERE TO FIND INFORMATION',
         CR,LF,LF,
         'DEFINING SUBSEQUENT CHANNEL ATTENTIONS (THE COMMAND BLOCK).',
         CR,LF,LF,
           '         TO SELECT ANOTHER MESSAGE TYPE Y-',EOT,EOT);

22  1    DECLARE    MSG5(*) BYTE
                DATA(CR,LF,ESC,E,
         '               8089 TASK COMMUNICATION PROTOCOL',
         CR,LF,LF,LF,
         '                ++++++++++++++++++++++++++++++++++++++++++++++++++',
         CR,LF,
         ' COMMAND BLOCK      +    BUSY FLAG        +    CHANNEL COMMAND WORD +',
         CR,LF,
         '  (ONE PER CHANNEL) ++++++++++++++++++++++++++++++++++++++++++++++++++',
         CR,LF,
         '                    +         PARAMETER  BLOCK  ADDRESS             +',
         CR,LF,
         '                    ++++++++++++++++++++++++++++++++++++++++++++++++++',
         CR,LF,
         ' PARAMETER BLOCK    ++++++++++++++++++++++++++++++++++++++++++++++++++',
         CR,LF,
         '                    +            TASK  BLOCK  ADDRESS               +',
         CR,LF,
         '                    ++++++++++++++++++++++++++++++++++++++++++++++++++',
         CR,LF,
         '                    +         USER DEFINED MESSAGE AREA             +',
         CR,LF,
         '                    ++++++++++++++++++++++++++++++++++++++++++++++++++',
         CR,LF,
         ' TASK BLOCK         ++++++++++++++++++++++++++++++++++++++++++++++++++',
         CR,LF,
         '                    +    TASK PROGRAM TO BE EXECUTED BY THE 8089    +',
         CR,LF,
         '                    ++++++++++++++++++++++++++++++++++++++++++++++++++',
         CR,LF,LF,
         '     AFTER A CHANNEL ATTENTION, THE 8089 READS THESE BLOCKS TO SEE WHAT THE',
         CR,LF,LF,
         'CPU WANTS (CHANNEL COMMAND WORD) AND WHERE TO FIND ADDITIONAL INFORMATION',
         CR,LF,LF,
         '(PARAMETER BLOCK). THE PARAMETER BLOCK GIVES THE TASK PROGRAM ADDRESS AND',
         CR,LF,LF,
         'PARAMETERS TO BE PASSED.      TO SELECT ANOTHER MESSAGE TYPE Y-',EOT,EOT);


23  1    DECLARE    INITTB(128) BYTE EXTERNAL;     /* TB TO INITIALIZE      */
                                                  /* 8251A & 8253          */

24  1    DECLARE    PROGTB(128) BYTE EXTERNAL;    /* TB FOR MESSAGE DISPLAY */
```

AFN01153A

Appendix A (Continued)


```
              /*************************************************************************/
              /*    THIS IS THE MAIN PROGRAM WHICH INITALIZES THE 8089 FROM RESET AND    */
              /* THEN ISSUES THE 89 A CA TO EXECUTE A TASK BLOCK WHICH INITALIZES THE    */
              /* 8251A AND THE 8253.  AFTER ALL INITALIZATION IS COMPLETE, THE PROGRAM   */
              /* IS TOTALLY INTERRUPT DRIVEN FROM THE 8089.  THE 8089 INTERRUPTS THE     */
              /* 8086 TO REQUEST A NEW MESSAGE FOR DISPLAY.   TO SERVICE THE INTERRUPT,  */
              /* THE 8086 TRANSFERS THE NEW MESSAGE FROM ROM TO THE MESSAGE BUFFER, SETS */
              /* UP THE APPROPRIATE TASK BLOCK PROGRAM AND ISSUES A NEW CA TO THE IOP TO */
              /* ALLOW IT TO DISPLAY THE NEW MESSAGE.  THE 8086 WILL HALT AFTER ISSUEING */
              /* THE CHANNEL ATTENTION AND WAIT FOR THE NEXT MESSAGE REQUEST.            */
              /*    AFTER EACH CA, THE 8089 WILL DISPLAY THE REQUESTED MESSAGE THEN POLL */
              /* FOR A NEXT MESSAGE REQUEST ENTERED AT THE CRT.  UPON RECEIVING A VALID  */
              /* REQUEST THE 8089 RETURNS THE REQUEST TO THE 8086, ISSUES AN INTERRUPT   */
              /* TO THE 8086 AND HALTS ITS CURRENT TB EXECUTION.  THE 8089 PERFORMS NO   */
              /* OTHER ACTIVITIES UNTIL AWAKENED BY THE CA FROM THE 8086 TO DISPLAY THE  */
              /* NEXT MESSAGE.                                                           */
              /*************************************************************************/

              /*************** MESSAGE PROCESSING INTERRUPT ROUTINE *******************/

  25   1      MSGDSPL:      PROCEDURE INTERRUPT 80 PUBLIC;

  26   2           IF  PB.CI='Y' THEN

  27   2               DO;
  28   3                    CALL MOVB(@MENU, @MSG$BUF,SIZE(MENU));
  29   3                    PB.LEVEL = FALSE;
  30   3               END;

  31   2           ELSE DO;

  32   3               PB.LEVEL = TRUE;
  33   3               DO CASE (PB.CI AND NMBR$MSK)-1;
  34   4                    CALL MOVB (@MSG1, @MSG$BUF, SIZE (MSG1));
  35   4                    CALL MOVB (@MSG2, @MSG$BUF, SIZE (MSG2));
  36   4                    CALL MOVB (@MSG3, @MSG$BUF, SIZE (MSG3));
  37   4                    CALL MOVB (@MSG4, @MSG$BUF, SIZE (MSG4));
  38   4                    CALL MOVB (@MSG5, @MSG$BUF, SIZE (MSG5));
  39   4               END;
  40   3           END;

  41   2           CALL MOVB (@PROGTB,@TB, SIZE (PROGTB));

  42   2           PB.TB$PTR = TB$89;
  43   2           PB.MSG$PTR = MSG$89;

  44   2           CB(0).CCW = DSP$CCW;
  45   2           CB(0).PB$PTR = PB$89;

  46   2           OUTPUT(CHAN$AT1)=00H;

  47   2           RETURN;

  48   2           END MSGDSPL;
```

AFN01153A

Appendix A (Continued)

```
PL/M-86 COMPILER    8089 PROTOTYPE DEMO                                                    PAGE   9


              /****************************** INITIALIZATION *****************************/

  49   1      START:    DISABLE;

  50   1           INTR$VEC$80 = @MSGDSPL;
  51   1           INTR$IP$80  = INTR$IP$80 - 27;

  52   1           OUTPUT(INT$STAT$PORT) = INT$ICW1;
  53   1           OUTPUT(INT$MASK$PORT) = INT$ICW2;
  54   1           OUTPUT(INT$MASK$PORT) = INT$ICW4;
  55   1           OUTPUT(INT$MASK$PORT) = INT$MASK;


  56   1           SINT.SYSBUS = SYSBUS$CMD;
  57   1           SINT.SCB$PTR = SCB$89;


  58   1           SCB.SOC = SOC$CMD;
  59   1           SCB.CB$PTR = CB$89;

  60   1           CB(0).BUSY = BUSYSTATUS;

  61   1           OUTPUT(CHAN$ATT) = 0;

  62   1           DO WHILE CB(0).BUSY = BUSYSTATUS;
  63   2           END;

  64   1           CALL MOVB(@INITTB,@TB,SIZE(INITTB));
  65   1           CB(0).CCW = INIT$CCW;
  66   1           CB(0).PB$PTR = PB$89;
  67   1           PB.TB$PTR = TB$89;
  68   1           OUTPUT(CHAN$ATT) = 0;

  69   1           ENABLE;
  70   1           DO WHILE TRUE <> FALSE;
  71   2           END;


  72   1      END PROTOTYPE$89;


MODULE INFORMATION:

     CODE AREA SIZE     = 196DH   6509D
     CONSTANT AREA SIZE = 0000H      0D
     VARIABLE AREA SIZE = 0000H      0D
     MAXIMUM STACK SIZE = 0022H     34D
     490 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-86 COMPILATION
```

AFN01153A

Appendix B

Appendix B

```
8089 ASSEMBLER                                                              PAGE   1
ISIS-II 8089 ASSEMBLER X004 ASSEMBLY OF MODULE DEMO89
OBJECT MODULE PLACED IN :F1:89DEMO.OBJ
ASSEMBLER INVOKED BY :F1:ASM89 :F1:89DEMO.SRC
                    1 ;*******************************************************************
                    2 ;*                                                                 *
                    3 ;*     ASM89 DEMONSTRATION PROGRAM FOR THE 8089 PROTOTYPE SYSTEM   *
                    4 ;*                                                                 *
                    5 ;*******************************************************************
                    6 ;
                    7 ;
                    8 NAME    DEMO89
0000                9 DEMO    SEGMENT
                   10 ;
                   11 ;
                   12 PUBLIC  INITTB
                   13 PUBLIC  PROGTB
                   14 ;
                   15 ;
                   16 ;**************************** EQUATES *****************************
                   17 ;
                   18 ;
 C000              19 DADDRESS_8251    EQU     0C000H
 C001              20 CADDRESS_8251    EQU     0C001H
 00CA              21 MODE_8251        EQU     0CAH
 0040              22 RST_8251         EQU     40H
 0025              23 COMMAND_8251     EQU     25H
 E003              24 MADDRESS_8253    EQU     0E003H
 0037              25 C0MODE_8253      EQU     37H
 E000              26 C0ADDRESS_8253   EQU     0E000H
 0016              27 C0_LSB_8253      EQU     16H
 0000              28 C0_MSB_8253      EQU     0
 0059              29 Y                EQU     59H
 0009              30 CI               EQU     9H
 5001              31 CHAN_CONTROL     EQU     5001H
 0004              32 MSG_POINTER      EQU     4H
 FF04              33 EOT_COMPARE      EQU     0FF04H
 0008              34 LEV              EQU     8H
 FF59              35 Y_COMPARE        EQU     0FF59H
 F837              36 MSG_COMPARE      EQU     0F837H
 FE37              37 SIX_SEV_COMPARE  EQU     0FE37H
 FF30              38 ZERO_COMPARE     EQU     0FF30H
                   39 ;
                   40 ;
                   41 ;********************** TASK1 - INTIALIZATION **********************
                   42 ;
                   43 ;
0000  3130 01C0    44 INITTB: MOVI     GB, CADDRESS_8251      ;INITIALIZE 8251A, COMMAND ADDRESS
0004  084D CA      45         MOVBI    [GB], MODE_8251        ;
0007  0000         46         NOP
0009  0000         47         NOP
000B  084D 40      48         MOVBI    [GB], RST_8251         ;SOFTWARE RESET
000E  0000         49         NOP
0010  0000         50         NOP
0012  084D CA      51         MOVBI    [GB], MODE_8251        ;2 STOP, CHAR LENGTH 7, X16
0015  0000         52         NOP
0017  0000         53         NOP
0019  084D 25      54         MOVBI    [GB], COMMAND_8251     ;REC AND TRAN ENABLED
```

AFN01153A

Appendix B (Continued)

```
8089 ASSEMBLER                                                                       PAGE   2

001C   3130 03E0    INIT53: MOVI    GB,MADDRESS_8253      ;INITIALIZE 8253, MODE ADDRESS
0020   084D 37              MOVBI   [GB],C0MODE_8253      ;CNT 0, MODE 3, BCD
0023   3130 00E0            MOVI    GB, C0ADDRESS_8253    ;COUNTER 0 ADDRESS
0027   084D 16              MOVBI   [GB], C0_LSB_8253     ;LSB = 16  8251A BAUD RATE GENERATION
002A   084D 00              MOVBI   [GB], C0_MSB_8253     ;MSB = 0
002D   0A4F 09 59           MOVBI   [PP].CI, Y            ;Y TO CI BYTE IN PARAMETER BLOCK
                                                          ;TO SELECT MENU FOR DISPLAY
0031   4000                 SINTR                         ;INTERRUPT 8086
0033   2048                 HLT                           ;WAIT FOR CA
                    ;
                    ;
                    ;***************** TASK2 - MESSAGE PROCESSING *********************
                    ;
                    ;
0035   D130 0150    PROGTB: MOVI    CC,CHAN_CONTROL       ;LOAD CHANNEL CONTROL WORD
                                                          ;MEMORY TO PORT
                                                          ;SYNC ON DESTINATION
                                                          ;GA SOURCE
                                                          ;TERMINATE ON MASK COMPARE
                                                          ;TERMINATE OFFSET = 0
0039   038B 04              LPD     GA, [PP].MSG_POINTER  ;MESSAGE POINTER, DMA SOURCE
003C   3130 00C0            MOVI    GB, DADDRESS_8251     ;8251A DATA ADDR, DMA DESTINATION
0040   F130 04FF            MOVI    MC,EOT_COMPARE        ;MASK COMPARE FOR EOT
0044   C000                 WID     16,8                  ;SOURCE BUS 16, DESTINATION BUS 8
0046   6000         DMA:    XFER                          ;START DMA VIA DRQ1 AND 8251A TXRDY
0048   5130 01C0            MOVI    GC,CADDRESS_8251      ;8251A COMMAND AND STATUS ADDRESS
004C   0AE7 08 14   LEVEL:  JZB     [PP].LEV, MSGSEL      ;CHECK LEVEL BYTE IN PARAMETER BLOCK,
                                                          ;MENU OR MESSAGE ?
0050   F130 59FF    MENSEL: MOVI    MC, Y_COMPARE         ;MASK COMPARE FOR Y
0054   28BA FD      RXRDY1: JNBT    [GC], 1, RXRDY1       ;RECEIVE READY ?
0057   08B5 FA              JMCNE   [GB], RXRDY1          ;Y ?
005A   0A4F 09 59           MOVBI   [PP].CI, Y            ;Y TO CI BYTE IN PARAMETER BLOCK
005E   084D 59              MOVBI   [GB], Y               ;ECHO
0061   8820 25              JMP     INTR86                ;
0064   F130 37F8    MSGSEL: MOVI    MC, MSG_COMPARE       ;MASK COMPARE FOR MESSAGE SELECT
0068   28BA FD      RXRDY2: JNBT    [GC], 1, RXRDY2       ;RECEIVE READY ?
006B   0091 02CF 09         MOVB    [PP].CI, [GB]         ;MESSAGE SELECTION TO CI BYTE
                                                          ;IN PARAMETER BLOCK
0070   0AB7 09 F4           JMCNE   [PP].CI, RXRDY2       ;0 THRU 7 ?
0074   F130 37FE            MOVI    MC, SIX_SEV_COMPARE   ;MASK COMPARE FOR 6 OR 7
0078   0AB3 09 E8           JMCE    [PP].CI, MSGSEL       ;6 OR 7 ?
007C   F130 30FF            MOVI    MC, ZERO_COMPARE      ;MASK COMPARE FOR 0
0080   0AB3 09 E0           JMCE    [PP].CI, MSGSEL       ;0 ?
0084   0293 09 00CD         MOVB    [GB], [PP].CI         ;ECHO
0089   4000         INTR86: SINTR                         ;INTERRUPT 8086
008B   2048                 HLT                           ;WAIT FOR CA
                    ;
008D                DEMO    ENDS
                    END
```

AFN01153A

Appendix B (Continued)

```
8089 ASSEMBLER                                                    PAGE    3


SYMBOL TABLE
------------

 DEFN VALUE TYPE  NAME
 ---- ----- ----  ----

   26  E000  SYM  C0ADDRESS_8253
   25  0037  SYM  C0MODE_8253
   27  0016  SYM  C0_LSB_8253
   28  0000  SYM  C0_MSB_8253
   20  C001  SYM  CADDRESS_8251
   31  5001  SYM  CHAN_CONTROL
   30  0009  SYM  CI
   23  0025  SYM  COMMAND_8251
   19  C000  SYM  DADDRESS_8251
    9  0000  SYM  DEMO
   79  0046  SYM  DMA
   33  FF04  SYM  EOT_COMPARE
   55  001C  SYM  INIT53
   44  0000  PUB  INITTB
   99  0089  SYM  INTR86
   34  0008  SYM  LEV
   81  004C  SYM  LEVEL
   24  E003  SYM  MADDRESS_8253
   83  0050  SYM  MENSEL
   21  00CA  SYM  MODE_8251
   89  0064  SYM  MSGSEL
   36  F837  SYM  MSG_COMPARE
   32  0004  SYM  MSG_POINTER
   69  0035  PUB  PROGTB
   22  0040  SYM  RST_8251
   84  0054  SYM  RXRDY1
   90  0068  SYM  RXRDY2
   37  FE37  SYM  SIX_SEV_COMPARE
   29  0059  SYM  Y
   35  FF59  SYM  Y_COMPARE
   38  FF30  SYM  ZERO_COMPARE

ASSEMBLY COMPLETE; NO ERRORS FOUND
```

AFN01153A